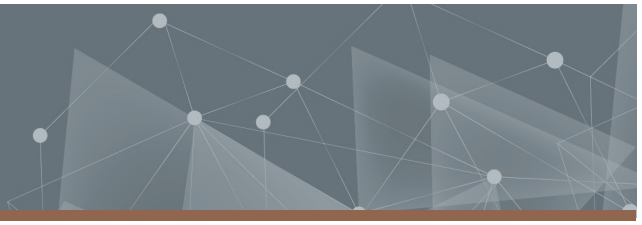
CHALMERS
UNIVERSITY OF TECHNOLOGY



Prediction of Vehicles' Movement based on Kalman-filter and LSTM Algorithms

Master's thesis

Implementing a prediction algorithm for the vehicles' movement

Master's thesis in Complex Adaptive Systems

AQSHAY CHANDRAMOULI

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Prediction of Vehicles' Movement based on Kalman-filter and LSTM Algorithms

Implementing a prediction algorithm for the vehicles' movement

Aqshay Chandramouli

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2021

Prediction of Vehicles' movement based on Kalman-filter and LSTM Algorithms
Implementing a prediction algorithm for the vehicles' movement
AQSHAY CHANDRAMOULI

© AQSHAY CHANDRAMOULI, 2021.

Supervisors: Elias Sevelin and Lan Wang, SCANIA
Examiner: Mats Granath, Department of Physics

Master's Thesis 2021
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2021

Prediction of Vehicles' Movement based on Kalman-Filter and LSTM Algorithms
To implement a prediction algorithm for the vehicles' movement
Aqshay Chandramouli
Department of Physics
Chalmers University of Technology

Abstract

The development of autonomous vehicles is advancing at a very high pace and could radically transform the future of the transportation system. If autonomous vehicles have to start driving safely and efficiently on public roads, they would have to know how the surrounding vehicles are behaving and act accordingly. This ability to predict and adapt is essential to make autonomous vehicles safe to use. In this thesis, the problem is considered as a sequence-to-sequence trajectory prediction problem. The proposed solution is a combination of a Kalman filter with an LSTM encoder-decoder framework. An attention mechanism is introduced to the LSTM framework for the interaction between the vehicles i.e. context attention and lane attention. Context vectors are chosen which helps in improving the accuracy. The model is trained and validated on the HighD Dataset and evaluated with the root mean squared error metric. The experiment shows good improvement in the accuracy when the attention mechanism is applied to the LSTM framework.

Keywords: Autonomous, LSTM, trajectory prediction, Kalman-filter, HighD, Neural Networks.

Acknowledgements

I would like to thank Elias Sevelin and Lan Wang for the guidance and supervision at Scania through the whole period. I would also like to thank Mats Granath for the suggestions and supervision at Chalmers University of Technology. I would like to thank Lianqiao Xiao for being a great partner and for all the suggestions and discussions. I would also like to thank everyone at the EADD group in Scania for all the help during the time. I also would like to express my gratitude to my family and friends who helped me throughout my masters.

Aqshay Chandramouli, Gothenburg, May 2021

Contents

List of Figures

List of Tables

1
Introduction

There is rapid advancement in the autonomous driving field. Soon, there will be autonomous vehicles on public roads. Due to the complex and dynamic driving environment, autonomous vehicles are tasked with many challenges. Autonomous vehicles need to anticipate future movement and adapt their behaviour accordingly. The vehicle needs to decide on whether to change lanes, cross intersections or overtake another vehicle. To do this, the vehicle needs to have an understanding of the future motion of nearby vehicles. Future trajectory prediction is not a deterministic task since there are differences between the driver's intentions and driving habits. This means that there are many possible solutions that can be taken in the same driving scenario. Utilising past information enables finding common solutions. There are existing path planning algorithms that depend on the estimation of future trajectories [**3**] [**37**] [**38**].

Most of the methods use an estimation motion state of the surrounding vehicle, with a kinematic model to predict future trajectories. Even though these methods are computationally efficient, it is still difficult to do long-term predictions using them. Manoeuvre-based models are an alternative to the kinematic models. These models learn the motion pattern of traffic from a trajectory dataset. They use recognition modules to predict a specific trajectory. The recognition uses the past information and motion states to give to the classifier. Random forest classifiers and RNNs have been used for manoeuvre-based models [**8**]. The major drawback of these models is that they are prone to poorly modelled safety-critical motion patterns which were not represented in the training dataset. Since motion prediction can be considered as a sequence classification task, many RNN-based classification methods have been proposed in recent times.

The LSTM framework consists of 3 components: encoder, context vector and decoder. Both the encoder and decoder work separately and have the ability to handle sequences of variable lengths. Encoder interprets the sequence and reduces the dimension of the sequence. Context vector is a single vector which summarises the entire input sequence. This vector acts like the input to the decoder. The first input of the decoder is initialized by using the context vector, finally it forecasts prediction.

The attention mechanism which is proposed in this thesis work is widely used in many deep learning tasks. The mechanism is inspired from the ability of humans to select relevant content from a large amount of information. So, the main idea is to choose a better context vector compared to the other context vectors in the

decoding process. The main task for the decoder is to extract the sequence information more intelligently. The main idea in the thesis is to implement two attention mechanisms for the vehicle trajectory prediction: lane attention and context attention mechanism. These two attention mechanisms are incorporated in the LSTM encoder-decoder framework.

1.1 Objective and Scope

The purpose of the thesis is framework design for future trajectory prediction based on past information. The focus is mainly on highway driving scenario. In this thesis, the primary focus is performance evaluation when the data is initially passed to the Kalman filter and the output is given as the input to the LSTM encoder-decoder framework. Two main approaches are evaluated by using the same LSTM framework. The first approach is to use the Kalman filter to filter the pre-processed dataset and send it to the LSTM framework. In the second approach, the framework is evaluated without using Kalman-filter to filter the dataset. The plan is to give 3 seconds trajectory history as the input and try to predict the next 3-5 seconds.

Furthermore, an attention mechanism was also incorporated into the LSTM framework. Two attention mechanisms that were planned to be added were the context attention mechanism and the lane attention mechanism. In context attention, we assume that the vector of the encoding process will have different trajectory prediction. So, in each time step prediction, the context vector will be re-weighted based on the original vector. In Lane Attention, the surrounding vehicles are divided into three groups based on their lanes and thus we get three context vectors. In the decoding process, we combine the context vectors as the final context vector. Another objective is investigation of the effects of the performance of the framework. The models proposed are also to be compared with a simple predictor model to see how well the proposed models outperform the simple predictor model.

1.2 Literature Study

Many approaches have been used for predicting the vehicle's movement. Surveys related to vehicle trajectory prediction can be found in [**5**] and [**6**]. These methods can widely be classified into three main categories: physical-based prediction, manoeuvre-based prediction and interaction-aware prediction. Most of the existing methods have two steps to solve the problem. The first step is to model the state and trajectory of the model. The model referred to here is mostly a kinematic model. The next step is to predict the future trajectories based on the model. The challenging part will be to define the proper model for the problem. A single model can be used for modelling trajectory history in the first step. We can represent all the features in a neural network to generate prediction results. In reality, it is difficult because a single model cannot capture all the variability in different situations.

In most machine learning algorithms, classification and regression are the two parts

of a technique. In context to vehicle prediction, classification consists of determining the driver's intentions and regression consists of predicting the future trajectory. Earlier, the motion of the vehicle was mainly described using kinematic models [**7**]. Most simple models assumed constant velocity, acceleration, turn rate, etc. These models are simple but not accurate enough for real road scenarios. More recently, manoeuvre based models were implemented. In this paper [**9**], a Kalman-filter is used to filter out the noisy sensor data to estimate the future position of the vehicle. In recent times, deep neural networks such as recurrent neural networks(RNN) and inverse reinforcement learning have also been introduced for vehicle trajectory prediction tasks [**10**].

A multilayer perceptron was used to predict the lateral motion of surrounding vehicles on a highway. The main concept used is that the vehicle drives within a lane except for lane change. Thus, an attempt was made to predict how likely the vehicle is, to choose a target lane when the target lane can be leftmost, centre or rightmost lane [**11**]. Bayesian filters and Support Vector Machines(SVMs) have also been used for the prediction of lane change intentions of the driver in highway scenarios [**12**]. The author proposes to use a combination of SVM and Bayesian filtering using features like speed, steering angle and lane change information. The problem is classified as a multiclass classification problem with the three classes being: 'left lane change', 'no lane change' and 'right lane change'. The Bayesian filter is used in the model to filter out the rate of false positives. Hidden Markov Models have also been used for the recognition of traffic scenarios. In [**13**], the authors propose a probabilistic approach where a traffic situation is characterised by modelling it as a distribution over a sequence using HMMs. The [**14**] proposes a trajectory prediction approach, where the history is used to get a joint probability distribution using a Gaussian Mixture Model. Thus, a probability distribution is given over the future motion. According to [**15**], the predicted trajectory can be represented in three versions. i) Estimate the deterministic future trajectory of the vehicle based on the model. However, there is always be some uncertainties in the model due to assumptions. ii) Predict uncertainties of the model in specific forms like normal distributions or a mixture of Gaussians. iii) To further generalize, Monte Carlo methods can be used to approximate the distribution by sampling randomly the input and trying to generate potential trajectories.

LSTMs have also been used to predict the temporal behaviour and predict the future trajectories of vehicles [**16**]. This experiment shows that it performs better than a Kalman-filter based prediction model. A directed predicted coordinate of the vehicles is obtained with the help of an LSTM network. An encoder-decoder structure for the LSTM is always better than the models discussed in the previous paragraph, since trajectory prediction takes in a data sequence as an input and the output is also a data sequence. The authors of [**17**] are some of the first group to try out the encoder-decoder structure for a sequence to sequence problem when the input and output were of different length.

Interaction-aware trajectory predictions are much better when compared to manoeuvre-

based trajectory predictions. Here [**18**], Deo et al. proposed a uniform framework for trajectory prediction. This includes a vehicle interaction module that considers the global context of surrounding vehicles. In [**19**], a system was designed to evaluate the possible dangers of trajectory predictions while taking only the interaction into context. Deo et al. proposed an LSTM encoder-decoder which uses convolutional social pooling compared to social pooling for learning interdependencies between vehicles. Here in [**21**], the group designed a social pooling mechanism to model the interactions between six surrounding vehicles. These interaction-aware models are extensive models for vehicle trajectory prediction. They are also reliable in long-term predictions.

One of the most important mechanisms in deep learning is the attention mechanism. In 2014, the Google DeepMind team proposed the attention model of visual attention. Xu et al. applied an attention mechanism in the task of image captioning. Attention mechanism has widely been used in various deep learning models based on neural networks. Several attention mechanisms for trajectory prediction have been proposed recently. In [**23**], a soft attention mechanism was introduced for the trajectory prediction of pedestrians.

1.3 Research Problem

The purpose of the thesis is to design a framework for vehicle trajectory prediction based on past information. The possible position of the target vehicle in defined future interval, is the expected final framework output. The framework is used on an open-source dataset called HighD dataset to do the prediction. The dataset should also be pre-processed in such a way that the right features are extracted. The features include past information about latitude, longitude, velocity and acceleration as well as information about the existence of nearby vehicles. The sampling rate for the data is 5 Hz. Therefore, we have 15-time steps of 3 secs and 25-time steps for the future 5 secs. The trajectory history of the target and eight surrounding vehicles are labelled as $X_i = X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9$. where X_1 is the target vehicle and the others are surrounding vehicles.

1.4 Methodology

The research problem of predicting trajectories will be approached by an LSTM encoder-decoder framework with an attention mechanism. The first 3 seconds are taken as the input to the encoder and the output will be the prediction of the next 5 seconds. Before the data is sent to the LSTM framework, it is sent to a Kalman filter which gives the estimation of the positions. This is given as the input to the encoder-decoder framework.
The final trajectory prediction model can be broken into following sub-objectives:

1. Preprocess the dataset by adding virtual surrounding vehicles and other features to ensure uniformity of the structure.

2. Design a Kalman-Filter to make an estimation for the preprocessed dataset.
3. Design a LSTM encoder-decoder framework with attention mechanism.
4. Evaluate the performance of the model.

2

Theory

In the following chapter we will go through the theory of artificial neural networks, recurrent neural networks and Long short-term memory(LSTM) and Kalman-filter will be explained.

2.1 Artificial Neural Networks

Artificial Neural Networks(ANN) have been taken as an inspiration from the biological neural network [**24**].Just like the way neurons act as connections in the brain to receive, transmit and process information via electric signals called synapses. The ANN act in a similar way. It is a network of interconnected units called artificial neurons, which learn by experience by adjusting its weight. It usually consists of fully connected layers, each having its own weight. The nodes are connected all the way from the input to the output via hidden layers. The weights are constantly updated via back propagation in the training process. The higher the weight for a node means that it has more significance for various characteristics. For example, if we take an image classifier it might contain information on the different shapes and features within the image. How the weights are updated depends on the number of epochs the network does on the training set. The size of the training set is also considered significant when the weight is updated in training process. Supervised learning is a type of machine learning task where the maps an input to an output based example input-output pairs. An optimal supervised learning task, classifies the class labels for unseen instances. A simple visualization of the neural network is shown in Fig 2.1. The four input nodes are coloured in green. The hidden layers are coloured in blue. In a more complex network, there can be several hidden layers. Those networks are called Deep neural networks. The hidden layers are connected to the output node.

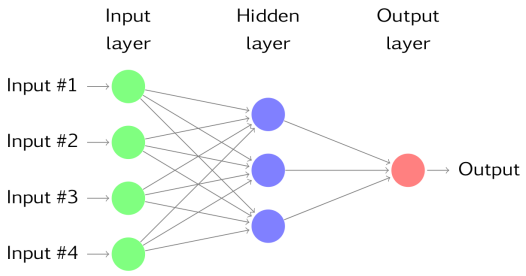


Figure 2.1: A visualization of a neural network [**25**]

2.1.1 Neuron

An artificial neural network contains processed units that are connected, which is called neurons. The artificial neuron receives message from other neurons and signals, and finally gives an output. The behaviour of the neuron is determined by the strength of the neurons connections to the other neurons.The artificial neurons are modelled by weighing different inputs to the neuron, the weights also evolve according to a particular learning rate. The neurons send a signal in the form of an activation function. The activation is modelled by a transfer function which determines firing rate of a neuron. The neuron receives signals from other neurons and finally transmits an output. The neuron is also modelled by a bias term in addition to the weights. The activation function is generally shifted using the bias term.

The mathematical formulation of output y of the neuron is, where x_n is the input of the nth neuron and w_b is the corresponding weight. φ is the activation function and b is the bias term.

$$y = \varphi(\sum_{n=1}^{N} w_n x_n + b) \tag{2.1}$$

The y is later compared to the true output y which shows how well the model behaves. Generally, they are compared with a loss function in the training phase and see how well the network learns.

2.1.2 Activation Function

In reality, the predicted data is most likely not linearly separable. But, when an input is multiplied in a weight matrix, the network is seen as a linear regression problem i.e. we try to fit the data with a linear function. So, activation functions are introduced to fit non-linearity in a network. They also have the ability to zero-out the neurons i.e. the neurons are either activated or they will be completely ignored. There are many activation functions, and the most basic one which is used in many deep learning researches is the sigmoid function [**27**].

Sigmoid function is a differentiable function, therefore the slope can be extracted with any two points. The output value from the sigmoid activation is fitted between the interval of [0,1]. The main disadvantage of the activation function is that it suffers from vanishing gradient problem, this can affect the flow of important feature information through the whole network during training. Vanishing gradient problem is a problem encountered when training neural networks. The problem is when the gradients will be very small which prevents the weights to be updated.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Another activation function is the tanh function which is a hyperbolic activation function. This activation function allows mapping of the negative weight values, where the inputs are squeezed in the interval [-1,1].

$$g(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

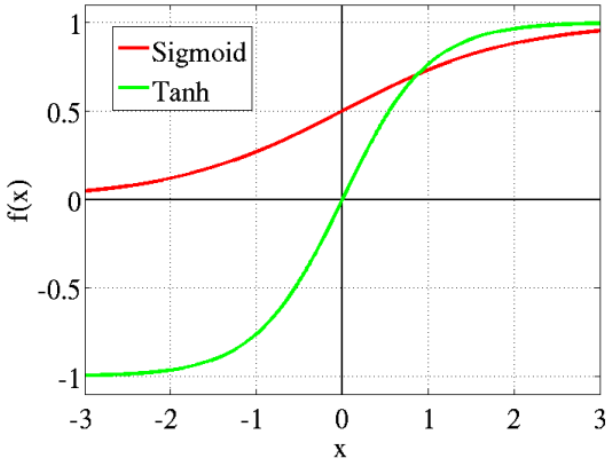


Figure 2.2: The form of sigmoid function and tanh function for input x

The activation function chosen is dependent upon the task as they all have advantages and disadvantages. The tanh function also suffers from vanishing gradient problem. But in comparison with sigmoid, it gives better accuracy when used for multilayer neural networks because it produces a zero centered output which supports the backpropagation process. It is also widely used in recurrent neural networks.

2.1.3 Training

The learning process of the algorithm refers to the phase where the weights and biases are updated [**30**]. Learning is mainly of two types: supervised learning and unsupervised learning. Supervised learning is used mainly for classification or a regression task where the output is known. The main goal is to find a function that can map input to the output. In unsupervised learning, the output is unknown. Here the main task lies in finding the structure of the data. The trajectory prediction in the thesis work is a supervised learning task as it is a regression task.

In the first stage of training, given a certain input the task of the ANN is to learn the output. The main aim when training is to minimise the error function that measures how well the training is going. The error function is chosen based on the application. In the thesis, mean squared error is used, which is defined as,

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_{k,true} = y_k)^2 \tag{2.3}$$

Here, y_k is the prediction of the training sample k and $y_{k,true}$ is the ground truth. By adjusting the weights and biases, the error function can be minimised. The most

fundamental method to update weights is through backpropagation. We can summarise backpropagation by saying that initially, data is passed forward through each layer and an output is obtained. Then the weight gradients and biases are calculated and the error is propagated backwards and optimized through an optimizer.

2.1.4 Optimization

The main goal during the training phase is to minimize the loss function. For deep learning tasks, stochastic optimizations are the most suitable optimizers. The equation shows how gradient descent technique is described. Here θ is the current point, γ is the learning rate which describes how weight, step is to be applied to the negative gradient. If the learning rate is slow then the convergance towards the minima is slow.

$$\theta_{i+1} = \theta_i - \gamma \frac{\partial J(\theta_i)}{\partial \theta_i} \tag{2.4}$$

A parameter update is performed for every training sample. The training sample is also shuffled randomly. This helps not get stuck in the local minima and this leads to eventually converging to the global minima.

One of the recent stochastic optimizers is ADAM (Adaptive Momentum Estimation). Due to its advantage on adaptive momentum and learning rate through the training phase, nowadays it is the standard optimizer for Deep neural networks.

2.1.5 Recurrent Neural Networks

Standard neural networks do not have the features properly if we are using sequential data. Traditional neural networks do not use the previous events in the data. Recurrent Neural networks can solve this issue, they have networks with loops with them which helps in the information to be persisted [**4**]. The nodes of RNNs pass information within themselves like a temporal sequence just like in the figure. There are not much difference with the structure of the RNN when compared with an ANN. It can be thought of as multiple copies of the same network each passing a message. This helps RNNs in dealing with time series datasets. Equation 2.5 shows how the repeating module of RNNs work.

$$h_t = tanh(W[h_t - 1, x_t] + b)t = 1, 2, ..., T \tag{2.5}$$

Here h_t is the output of the network at timestep t, x_t is the input data at time step t and b is the bias. W is the transformation matrix. The image shows the structure of the recurrent neural network when it is unfolded. The same weights are used in every time step.

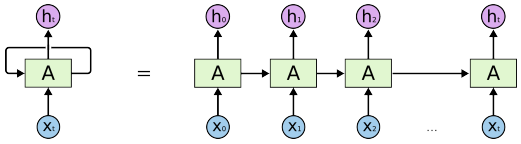


Figure 2.3: Unrolled Recurrent Neural Network

RNNs have been used for many problems successfully: speech recognition, language modeling, translation, image captioning and other things. They are many different variations of the RNNs, and one of the most successful RNN is the LSTM, introduced by Hochreiter and Schmidhuber [**4**].

2.2 Long Short-term Memory

Sometimes we only need to look into the recent information to perform the task. In cases where the gap between the relevant information is small, RNNs can learn to use the previous information.To overcome the problem of vanishing gradient, Long short-term memory networks can be used. It is capable of learning long term dependencies. Their default behaviour is to remember information for long-term. The chain like structure is similar to recurrent neural networks, but the repeating module has a different structure.

2.2.1 Architecture

The architecture of a LSTM is similar to a RNN, but instead of summation units, it has LSTM units. A block consists of three multiplicative units called Gates. The gate consists of a sigmoid activation function followed by a pointwise multiplication. It mainly used to control the flow of information i.e. save and access the information over a long period of time helping in solving the vanishing gradient problem. The cell state in the LSTM represents the memory.The figure shows a single LSTM cell. The whole rectangle is the "cells". Several parts make up the LSTM cell.

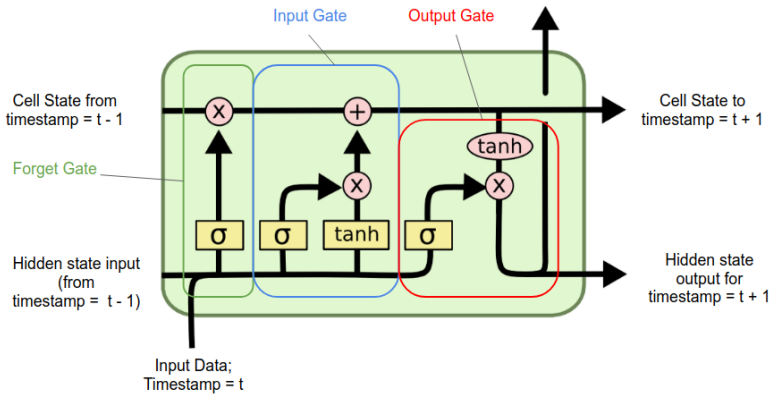


Figure 2.4: A single LSTM Cell

The main parts which make up the cell are:

1. The Cell State
2. The Hidden State
3. The Gates

2.2.1.1 Cell State

The Cell state is used to encode the data from the previous time steps that have been processed. In this state, when each time a time-step of data passes a copy of the data is filtered in the forget gate and another copy is sent to the input gate. Thus, new information is received by it in the input gate and it forgets through the forget gate. The memory of the cell state in the current time-step c^t gets passed into the next time-step. The current time-step is usually defined as,

$$c_t = f_t.c_{t-1} + i_t.c_{-t} \tag{2.6}$$

c^t is a combination of the sigmoid and tanh activation functions. It runs straight through the entire chain with only minor linear activations. The sigmoid saturates the signal to values between 0 and 1 while the tanh saturates the signal to values between -1 and 1.

2.2.1.2 Forget Gate

The forget gate decides how much information should be forgotten. It is defined as,

$$f_t := \sigma(w_f[x_t, h_{t-1}] + b_f) \tag{2.7}$$

f^t is the result obtained from the sigmoid function of the weighted input and the previous hidden state h^{t-1}. This signal gets to the input gate and output gate. The reason it is passed through sigmoid is because any value which get multiplied by zero is zero. If the forget gate outputs a matrix of values which are close to 1, then the forget gate concludes that the information is very relevant for the next time-step.

2.2.1.3 Input Gate

The input gate does the feature extraction to encode the data which is meaningful to the LSTM and another time to see whether hidden state and data from the current time-step should be remembered.

$$i_t := \sigma(w_i[x_t, h_{t-1}] + b_i) \tag{2.8}$$

$$c_t := tanh(w_c[x_t, h_{t-1}] + b_c) \tag{2.9}$$

The input signal can be defined as the multiplication of both the equations. The i_t decides the values to be updated and c^t gives the possible values which can be added.

2.2.1.4 Output Gate

The output gate uses the same concept as encoding and sampling to incorporated into the cell state. It specifically determines the next hidden state. It is defined as,

$$h_t := o_t.tanh(c_t) \tag{2.10}$$

The output gate attempts to capture the most important cell state information. This is passed into the tanh function.

To summarise the whole process, the cell state serves as a memory, the forget gate decides which information is necessary, the input gate decides which information is relevant from the current input and the output gate determines the next hidden state.

2.3 Encoder-Decoder Structure

In this thesis, the prediction problem is considered as a sequence-to-sequence problem.The information which is required can be viewed as a sequential information. For sequence to sequence problem, the most common method used is the encoder-decoder framework. The data is a time sequence of features extracted from historical vehicle data. LSTM encoder-decoder structures were first introduced for machine translation, these are used to solve the trajectory prediction problem. These frameworks help to solve sequence-to-sequence problems, especially when the size of sequence length of the input and output is different. The below figure shows the encoder-decoder framework.

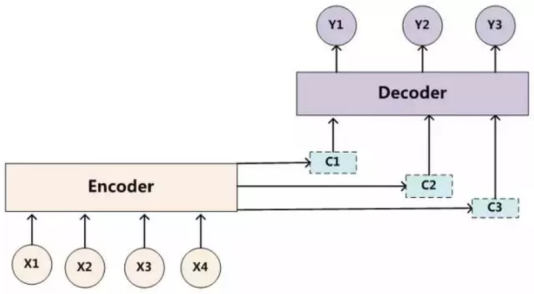


Figure 2.5: An LSTM encoder-decoder framework

As the name suggests, it contains two parts:
Encoder is a stack of recurrent units, where each unit takes in one element of the input sequence [**17**]. Information is passed along the loop module one step at a time. The final hidden node which collects all the information in the input sequence, is saved as a context vector as represented in figure 2.6, where the length is decided by the number of nodes in the encoder layer.
Decoder is also a stack of recurrent units where each predicts the output sequence. Hidden states are passed along the time sequence. The first hidden state is the output of the encoder. Different context vector (C1, C2 and C3 in figure 2.6) is chosen based on each time step. We can see that the input sequence and output sequence are correlated. GRU units and LSTM are used in the encoder-decoder framework. The attention mechanism is fitted into the model to overcome long sequence problems.

2.4 Attention Mechanism

Attention mechanism has changed the way Deep Learning Algorithms work. It has spawned the rise of recent breakthroughs in Natural Language Processing (NLP). Just like how a neural network is considered to mimic the actions of the human brain in a simplified way, attention mechanism is also an attempt to implement the same action of concentrating only on a few relevant things while ignoring the other information.

Before attention model was introduced in 2015 [**31**], most of the methods were implemented based on encoder-decoder RNNs/LSTMs. The main drawback of this approach was long range dependency issue. RNNs cannot remember long sequences as discussed before. While the LSTM solves the problem, it tends to be forgetful in some specific cases. Here, the bidirectional LSTM generates a sequence of annotations $h_1, h_2, .., h_{Tx}$. The last state of the encoder h_{Tx} was used as the context vector. The context vector for the output is generated using the weighted sum of the annotations.

2.4.1 Soft Attention

The most common method for a sequence to sequence problem is the encoder-decoder framework. The framework mainly consists of 3 main components: encoder, context vector and decoder. The context vector is a very important element in the framework. The input sequence information is stored in the context vector, which is used for prediction of the position by the decoder. There are three elements needed for the position prediction i.e. hidden state h_t, the prediction of the last time step y_{t-1} and the context vector.

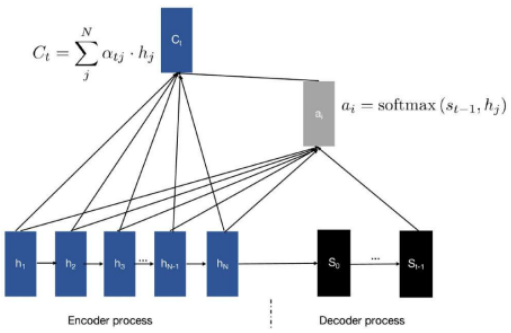


Figure 2.6: Soft attention mechanism

The soft attention context vector C_t can be computed as a weighted sum of the hidden states.

$$C_t = \sum_{j=1}^{15} a_{ij} h_j \tag{2.11}$$

The weights a_{ij} is calculated using a softmax function by the following equation,

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.12}$$

e_{ij} is the output of the feedforward neural network described by the function a. If the enocder produces T_x number of hidden state vectors each having dimension d, then the input dimension of the network is $(T_x, 2d)$ where the dimensions of the previous state of the decoder is also d. On the e_{ij} a tanh function is applied followed by a softmax.

$$y_t = FC(LSTM(y_{t-1}, h_{t-1}, C)) \tag{2.13}$$

The encoder-decoder framework has its own limitations. The context vector cannot completely represent the entire sequence information and there are chances that the first sequence information might be diluted or overwritten by the sequences that follow. Hence, the idea of soft attention mechanism is to choose a suitable context vector for each time step of the prediction.

2.4.2 Context Attention

Soft Attention mechanism is not suitable for vehicle trajectory prediction because it pays more attention to the order between input sequence and output sequence. In driving scenarios, more attention need to paid to the spatial and temporal information like relative speed and distance to the surrounding vehicles [**36**]. In Context attention, h_{15} is assumed as the most important hidden state. The main reason is because h_{15} includes all the driving scenario information for the past 3 seconds. The weighted sum is not the context vector here.

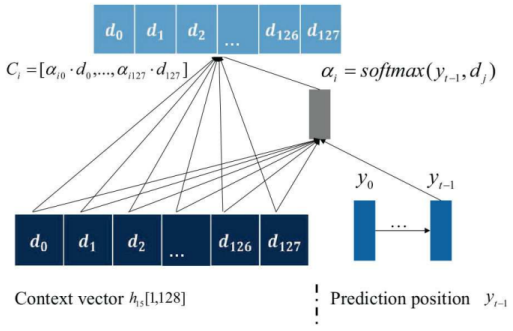


Figure 2.7: Context attention mechanism

As seen in figure, h_{15} is a vector size of [1,128]. These 128 elements represent the features of the nine vehicles. But the importance of each surrounding vehicle with the respect to the trajectory of the target vehicle. Therefore the elements in h_{15} are re-weighted based on the surrounding vehicle's importance. For each time step, the context vector is calculated. Then, to calculate the weights, a light neural network is designed which contains two fully connected layers and one softmax layer.

2.4.3 Lane Attention

Lane attention is a prototype of spatial attention in lane [**36**]. In this attention mechanism, the nine vehicles are divided into three groups based on their lane positions i.e. one group for vehicles in the left lane, one group for vehicles in the current lane and one group for vehicles in the right lane. Vehicles in different lanes have different influence on the target vehicle. For example, if the target vehicle changes from current lane to right lane then the left most lane can be ignored as the current lane and right most lane are more important.

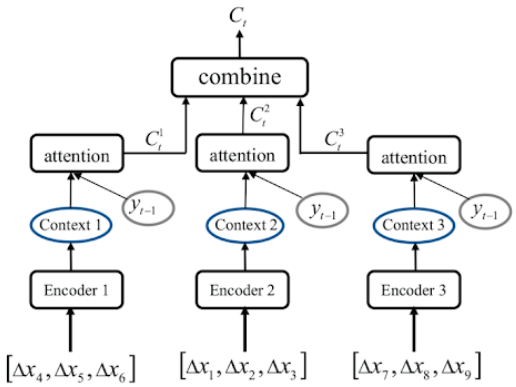


Figure 2.8: Lane attention mechanism

Each vehicle group has its encoder network. So, for three encoders there are three context vectors. The final context vector is a weighted combination of the three original context vectors. The importance of each original context vector is determined by the relative location to the target vehicle. y_{t-1} is the prediction position of the last time step in the decoding process. The format of the final context vector is,

$$C_i = [a_{i1}.context1, a_{i2}.context2, a_{i3}.context3] \tag{2.14}$$

2.5 Kalman-Filter

Kalman Filter is a recursive filter based on noisy measurements and estimates of state dynamics [**9**]. It is used to estimate system parameters by estimating a joint probability distribution over the variables for each time frame. The algorithm works in a two-step process: the prediction state and the update state. In the prediction state, the Kalman filter produces estimates for the current state variables along with other uncertainties. After the outcome of the measurement is known, the estimates are updated using weighted average. The filter uses known control inputs to the system and multiple sequence measurements to form an estimate of the system's varying quantities which is better than just using one measurement. The covariance matrix and state estimates are used to handle multiple dimensions involved in a single set of calculations.

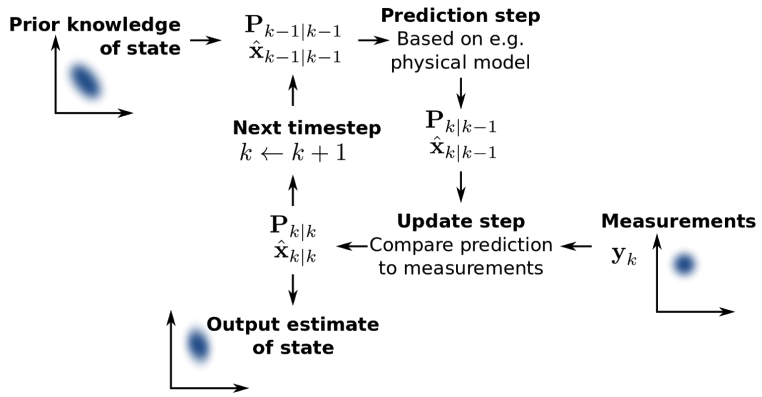


Figure 2.9: Kalman Filter Process

The kalman filter keeps track of the estimated state of the system and variance of the estimate. $\hat{x}_{k|k-1}$ denotes the estimate of the system's state at a time step k. $P_{k|k-1}$ is the corresponding uncertainty.

2.5.1 Process to Estimate

The kalman filter addresses the problem of trying to estimate the state $x \in \Re^n$ of a discrete time controlled process. The linear stochastic difference equation,

$$x_{k+1} = A_k x_k + B u_k + w_k \tag{2.15}$$

The measurement $z \in \Re^m$ is given as,

$$z_k = H_k x_k + v_k \tag{2.16}$$

The variables w_k, v_k are the process and measurement noise respectively. They are assumed to be independent of each other with normal probability distribution. The matrix A relates the state at time step k to state at time step k+1. The matrix B relates to control input $u \in \Re^l$. Finally, the matrix H realtes to the measurement z_k. All the matrices in this process depend on k, the Kalman gain.

2.5.2 Computational Origin of the Filter

The priori state estimate at time step k is defined given the knowledge of the process prior to step k, and $\hat{x}_k \in \Re^n$ to be the posteriori state estimate at time step k for given measurement z_k. The estimate errors are defined as,

$$e_k^- \equiv x_k - \hat{x_k^-} \tag{2.17}$$

$$e_k \equiv x_k - \hat{x_k} \tag{2.18}$$

Now, the priori estimate and posteriori estimate error covariances are,

$$P_k^- = E[e_k^- e_k^{-T}] \tag{2.19}$$

$$P_k = E[e_k e_k^T] \tag{2.20}$$

When the equations are derived, the main goal is to find an equation which can posteriori state estimate as a linear combination of the priori estimate and the weighted difference the actual measurement and measurement prediction as shown in,

$$\hat{x_k} = \hat{x_k^-} + K(z_k - H_k\hat{x_k^-}) \tag{2.21}$$

If the residual is zero, then it means that the two are in complete agreement. The matrix K is the gain or the blending factor which minimizes the posteriori error covariance.

2.5.3 Kalman Filter Algorithm

The kalman filter estimates the process by a feedback control: the process state is estimated by the filter and a form of measurement is obtained. The equation falls into two groups: time update group and measurement update equations. The time update can be thought as predictor equations while measurement equations can be thought of as corrector equations.

The first task in the measurement step is to update the Kalman Gain K_k. The next step is to measure the the process to obtain z_k and then to generate as a posteriori state estimate by incorporating the measurement.

After each update and measurement pair, the process is repeated with previous posteriori estimates used to predict new priori estimates.

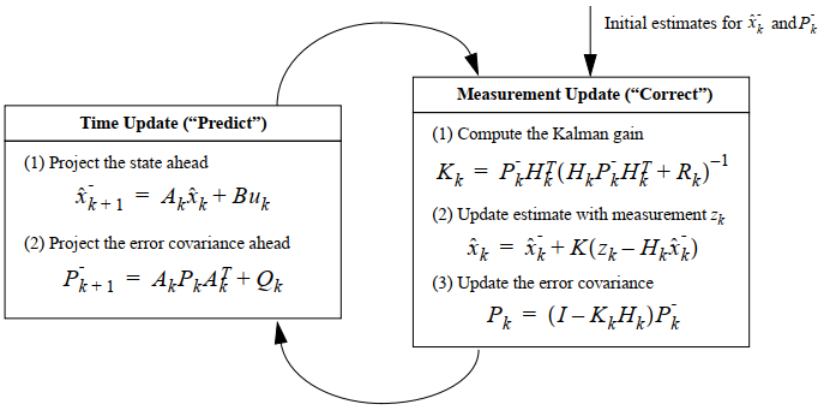


Figure 2.10: The Algorithm of the Kalman Filter

Each of the measurement error covariance matrix R_k and the process noise Q_k might be measured prior the operation to the filter. In the case of R_k, we need to be able to do this to measure the process in order to determine the variance of the measurement error. But in the case of Q_k, the choice is less deterministic.

In cases where the Q_k and R_k are constants, both the estimation error covariance and Kalman Gain will remain constant.

3

Methods

3.1 Dataset Introduction

The encoder-decoder framework needs a vehicle trajectory dataset to create an environment to train the network. The framework will be evaluated in the HighD dataset which is a naturalistic vehicle trajectory of vehicles at the German highways. Most of the trajectory predictions methods are done by using the NGSIM dataset [**34**]. However, the quality of NGSIM is not very good and some of the features which we get in the HighD dataset are missed.

3.1.1 HighD Dataset

HighD dataset is used which was published in 2018 [**1**]. The dataset is a naturalistic vehicle trajectory recorded on a German highway around Cologne with drones. Typical limitations of established traffic data collection methods such as occlusions are solved by aerial perspective when using drones. It was recorded at six different locations and 60 recordings were made covering a road segment of 420 m lengths with a total duration of 16.5 h. The dataset quality is better compared to other vehicle trajectory datasets.

Using state of the art computer vision algorithms, more than 110,500 vehicle's trajectories are extracted. A vehicle's trajectory includes vehicle type, size, velocity and information related to the position. It also contains some pre-extracted information which contains surrounding vehicles, time changes and lane changes. A total of 333,140 driving sequences can be extracted and each sequence contains 8 seconds of driving information. There are a total of 107,791 lane-change sequences and 225,349 lane-keep changes. The lane-change sequence is when the vehicle changes lane and the lane-keep sequence is where the vehicle does not change lane. The dataset is split in a way that 80% is used for training while 10% is used for validation and the rest 10% used for training. The figure shows the trajectory figure for the HighD dataset. We can see that the bounding boxes match exactly the shape of the vehicle.

Figure 3.1: HighD Trajectory figure

3.2 Entire Framework

The architecture of the entire framework is displayed in the below figure. It is mainly comprised of two parts: the Kalman filter and the LSTM encoder-decoder. The process of the framework can be split into two phases: training and prediction. We can consider the data preprocessing and the tuning of the network structures and parameters as part of training.

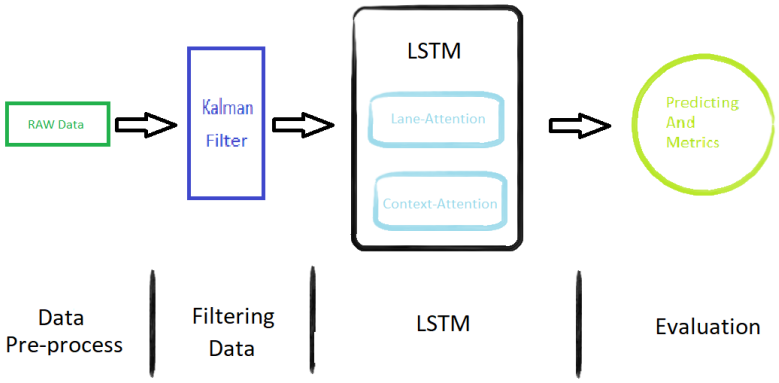


Figure 3.2: Entire Framework

During training, first, the data preprocessing is done. For deep learning neural networks, the quality of the dataset is important for performance. Most processes have been done in the preprocessing step. Some of the steps include adding virtual surrounding vehicles and data resampling. The final data contains information about the positions, velocity and information of the surrounding vehicles. After the data preprocessing is done, the data is given as input to the Kalman filter. Kalman filter is an optimal state estimator which iterates between the prediction state and update state. Kalman filter estimates the future trajectories which are given as input to the encoder LSTM. The input dimension of input of encoder is 36. The input is the

estimated position and velocity of the nine vehicles in the past 3 seconds. These will be encoded step-by-step into a context vector. The context vector is given as input to the decoder. The decoder will generate the prediction for the next 5 seconds. After the future trajectories are predicted, a loss function is pertinent to the performance of the model. Root Mean Square is the loss function that is used. Details of each part of the framework will be given in the following sections.

3.3 Data Preprocessing

It is necessary to do preprocess the HighD dataset. Preprocessing is a complex task involving many details. The main things to be handled in the dataset involves data integrity and handle problem of imbalanced data.

3.3.1 Highway Scenario

The figure shows a simple representation of the highway scenario. The blue vehicle represents the target vehicle and the lines show the possible trajectory in the next few seconds. The black vehicles represent the surrounding vehicles that will influence the movement of the target vehicle. The dataset is preprocessed in a way that the 8 surrounding vehicles will always be present either as the preceding or following vehicle in the same lane and also in the surrounding lane.

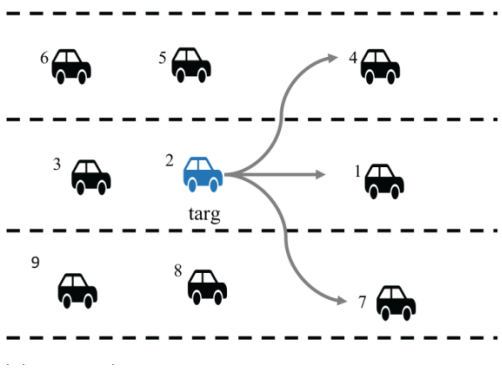


Figure 3.3: Highway Driving Scenario

The driving scenario of the past 3 seconds is used to predict the next 5 seconds. The data sampling rate is 5 Hz. So, for the input there will be 15 time points and the output will have 25 time points.

The trajectory of the target vehicle and the surrounding vehicles is defined as $X_i = X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9$. The trajectory of vehicle i is defined as $X_i^t = (x_i^t, y_i^t, v_{xi}^t, v_{yi}^t)$ where x and y are the longitudinal position and latitudinal position respectively. v_x and x_y are the longitudinal velocity and transverse velocity respectively. The axes x and y represent direction of motion on the highway and direction perpendicular to it respectively.

Relative position and relative speed is more important compared to the absolute position when performing driving decisions. So, for surrounding vehicles relative information to the target vehicle is chosen over absolute position and speed.

3.3.2 Labelling

The HighD dataset includes data extracted from 60 different recordings. Four files are provided: an image of the recorded highway, a csv file describing recording location data, one containing an overview of the recorded vehicle tracks and csv file for the tracks' trajectories [**33**].

Figure 3.4: Image of the Highway section

The above image was created from the recording by removing all the vehicles using a filter. For each recording, an image of the highway section is added.

Tracks csv files contains all time dependant values for each track. It contains information such as current velocities, positions and information about the surrounding vehicles. The tracksMeta file contains an overview of all the tracks. The purpose of the file is to filter tracks based on driving direction or class.

Initially, the tracks csv files and tracksMeta csv files are loaded as loaded as dataframes using Pandas Library. The tracksMeta contains information about the 'drivingDirection', 'numLaneChanges'(number of lane changes detected by changing lane id), 'class'(car or truck) and 'numFrames'(total lifetime of the track as number of frames). These information are taken from the meta data and added to tracks data. After this labelling is done.

Dataset is restructured to make implement the driving scenario. So when we select a particular vehicle then we can calculate the vehicles which are preceding and following in the same lane. Then these are also found for the right lane and the left lane. We also find the vehicles which are along side the target vehicle in the left lane and right lane

3.3.3 Adding virtual vehicles

We select eight surrounding vehicles for the target vehicle. The input of the model information about the position and velocity is given of the eight surrounding vehicles and the target vehicle. In reality, not all the eight surrounding vehicles exist in all the scenarios. To ensure uniformity to the structure, virtual vehicles are added for the missing surrounding vehicles.

```
if line.precedingId==0:
    line.prexVelocity=line.xVelocity
    line.preyVelocity=line.yVelocity
    line.prex=line.x+200
    line.prey=line.y
if line.followingId==0:
    line.folxVelocity=line.xVelocity
    line.folyVelocity=line.yVelocity
    line.folx=line.x-200
    line.foly=line.y
if line.leftPrecedingId==0:
    line.lprxVelocity=line.xVelocity
    line.lpryVelocity=line.yVelocity
    line.lprx=line.x+200
    line.lpry=line.y
if line.leftAlongsideId==0:
    line.lalxVelocity=line.xVelocity
    line.lalyVelocity=line.yVelocity
    line.lalx=line.x+150
    line.laly=line.y
if line.leftFollowingId==0:
    line.lfoxVelocity=line.xVelocity
    line.lfoyVelocity=line.yVelocity
    line.lfox=line.x-200
    line.lfoy=line.y
```

Figure 3.5: Part of the code used for adding virtual vehicles

The figure shows the part of the code which is used to add virtual vehicles to the scenario. We select a target vehicle and see if there are any preceding and following vehicles and also vehicles which are along the target vehicle either in the left side or right side.

After the virtual vehicles are added, then the data is split into track input and the position output. Adding the virtual vehicles to the scenario ensures that the data is uniform. Uniformity is ensured as it makes sure the training of the model is done properly.

TargetVehicle	'id','datataid','x','y','xVelocity','yVelocity'
Following	'folx', 'foly','folxVelocity','folyVelocity'
Preceding	'prex', 'prey','prexVelocity','preyVelocity'
RightVehicle	'ralx', 'raly','ralxVelocity','ralyVelocity'
RightFollowing	'rfox', 'rfoy','rfoxVelocity','rfoyVelocity'
RightPreceding	'rprx', 'rpry','rprxVelocity','rpryVelocity'
LeftVehicle	'lalx', 'laly','lalxVelocity','lalyVelocity'
Leftfollowing	'lfox', 'lfoy','lfoxVelocity','lfoyVelocity'
Leftpreceding	'lprx', 'lpry','lprxVelocity','lpryVelocity'

Figure 3.6: Ideal lane-based scenario after preprocessing

The above figure shows the ideal lane-based scenario after preprocessing steps are done. In the thesis, the main delimitations that are considered are: the shape of the vehicle is disregarded and considering only the velocity and positions as the inputs and predicting the position.

3.4 Trajectory Prediction

To consider the problem of trajectory prediction, Figure 3.3 is considered. As shown in Figure 3.3, a highway scenario is considered where the blue vehicle is the target vehicle. By considering the trajectory history of the eight surrounding vehicles(black vehicles) and the target vehicle(blue vehicle), the future trajectory of the target vehicle is predicted. The data sampling rate is 5 Hz.

3.4.1 Problem Formulation

Two different approaches are used for predicting the trajectories. But the general problem considered is to determine the trajectory of the target vehicle by considering a sequence of previous information. We want to predict the next 5 seconds when 3 seconds is given as input. The first approach is to preprocess the data and then give it as input to the encoder. The other approach is where a Kalman filter is used to estimate the latitudinal and longitudinal positions. The Kalman filter is used on the entire dataset before the virtual vehicles are added. After using the Kalman filter, the virtual vehicles are added. It is implemented using the Pykalman library. After the positions are estimated, the data is given as input to the LSTM encoder. Both the approaches are compared to see whether using a Kalman filter for estimation brings out better prediction results.

3.4.2 Hyper-parameters

For the encoder-decoder framework, the input dimension is 36 and the output dimension is 2. The dimensions of the hidden layers is 128. A two layer unidirectional LSTM is used. The number of epochs is set to 100. Optimiser used is Adam optimiser where the learning rate is set to 0.1 and weight decay is set to 0.00001. The number of layers for encoder and decoder are set to 2. The hidden dimension is set to 128 and the encoder and decoder hidden dimensions are set to 64 and 16 respectively. The initial weights are given by gaussian distribution between [-15,15]. An early stopping mechanism is used to avoid overfitting problems. The number of layers for the encoder and decoder is set to 2. The loss used to calculate the training loss, validation and testing loss is MSE Loss.

3.5 Comparison between the models

To analyse the different attention mechanisms on the model and to compare the results four different trajectory prediction methods are proposed. The models are implemented using Pytorch Library. It is also made sure that the encoder and decoder have the equal amount of layers. The teacher forcing ratio is the probability to use the ground truth in the prediction. Here the ratio is set to 0.5.

3.5.1 encoder-decoder without attention

This model is the baseline model where attention is not given. The trajectories predicted by this model is used as the baseline to see how implementing attention mechanism on the model helps in trajectory prediction.

3.5.2 Soft attention based encoder-decoder

Soft Attention is added to the LSTM encoder-decoder framework. An unique context vector is chosen in each time step in the decoding process. The order of the input and output sequence is given more consideration. Context vector gives more attention in the model to the correspondence between the input and output sequence. Attention is applied using the Pytorch library.

3.5.3 Context attention based encoder-decoder

In this model, the spatial relationship between the target vehicle and surrounding vehicles are considered. The model gives more consideration to information like distance between the surrounding vehicles and relative speed. The context vector are calculated to the given different weights to the elements to the original context vector h_{15}.

3.5.4 Lane attention based encoder-decoder

This model focuses on the spatial location information. The model is a prototype of spatial attention based on lane. The vehicle are divided according to their lanes. These vehicle groups are encoded separately. The idea for grouping is inspired from the experience of human drivers. The context vectors of the three groups are combined for each time step in the decoding process. The importance of the context vector is based on the location of the target vehicle.

3.6 Evaluation

In this thesis, the future trajectories are predicted over a 5 seconds. To compare the performance of the models in short-term and long-term predictions, the prediction results are recorded from 1 to 5 seconds. The predictions are evaluated against the ground truth. The metric which is used is Root Mean Squared Error(RMSE). This metric is commonly used in trajectory forecasting tasks. The formula can be described as,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((\hat{x} - x)^2 + (\hat{y} - y)^2)} \tag{3.1}$$

where $\hat{x}, \hat{y}$ and $x.y$ are the prediction and true value of the longitudinal and lateral positions respectively. A new loss function is designed based on the standard RMSE.

The idea is to give more penalisation to the lateral position error.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((\hat{x} - x)^2 + 3(\hat{y} - y)^2)} \tag{3.2}$$

The error for the lateral position is tripled compared compared to the longitudinal position error. Also to evaluate the models more comprehensively, lateral position and longitudinal position errors are calculated separately.

4
Results

The results based on section 3.6 will be compared in this chapter. Apart from using the RMSE evaluation metric, the predicted trajectory of the models is also plotted with the true trajectory, to see how accurate the trajectory is. The models will be compared, with and without using Kalman filter in the dataset.

4.1 Trajectory Prediction without Kalman Filter

The evaluation of the model is done using RMSE. The table 4.1 shows the RMSE values for the model being compared over a 5 second prediction horizon. The values are in meters. The performance of the lane attention model is better in long-term of 5 seconds. The model with soft attention is similar to the baseline model and does not improve significantly as compared to the other two attention models. The attention models also perform well for longer predictions. Comparing lane attention and context attention models, the lane attention performs better as the prediction error is 1.71 m. The main reason for the performance is because lane attention mechanism gives importance to the lane information for the vehicle trajectory prediction. There is not a big difference for all the models for short term prediction i.e. in the 1st and 2nd prediction horizon. A significant change is also seen from the 4th prediction horizon to the 5th prediction horizon. From the three tables, the lane attention model and context attention model perform well in 5 seconds horizon. Overall, the RMSE value changes from 0.95 m at the first second to 1.71 m in the 5th second. For context attention model, the change is from 1.32 m to 1.85 m. The error for the two models does not increase significantly as the number of seconds increases which shows it is good for longer trajectory predictions.

Model	1s	2s	3s	4s	5s
Lane Attention	0.74	0.74	0.78	0.92	1.18
Context Attention	0.86	0.86	0.93	1.10	1.35
Soft Attention	1.14	1.18	1.30	1.55	1.88
Baseline	1.05	1.08	1.19	1.45	1.80

Table 4.1: Position RMSE(m)

In tables 4.2 and 4.3, longitudinal error and lateral error are also calculated to analyse the model. In terms of longitudinal position, the lane attention model gets

a better result compared to other models with prediction error of 1.67 m in the 5th second prediction horizon.

Model	1s	2s	3s	4s	5s
Lane Attention	0.73	0.72	0.75	0.88	1.12
Context Attention	0.84	0.85	0.90	1.06	1.30
Soft Attention	1.17	1.13	1.25	1.49	1.81
Baseline	1.02	1.14	1.36	1.39	1.74

Table 4.2: Longitudinal Position RMSE(m)

In the lateral direction, the RMSE of the context attention model is changed from 0.31 m to 0.41 m.

Model	1s	2s	3s	4s	5s
Lane Attention	0.15	0.15	0.19	0.25	0.33
Context Attention	0.17	0.18	0.21	0.27	0.35
Soft Attention	0.37	0.34	0.38	0.42	0.50
Baseline	0.33	0.33	0.34	0.40	0.48

Table 4.3: Lateral Position RMSE(m)

Plot is made of the trajectory for each instance namely lane keep, lane change to left and lane change to right. The plot is made with x and y positions. The first plot shows 25 time steps and each point in the second plot shows the one second of prediction. It is used to compare the prediction of the models to the ground truth. The trajectories predicted by the model for lane keep scenario shows that lane attention has high accuracy and context attention has low accuracy.

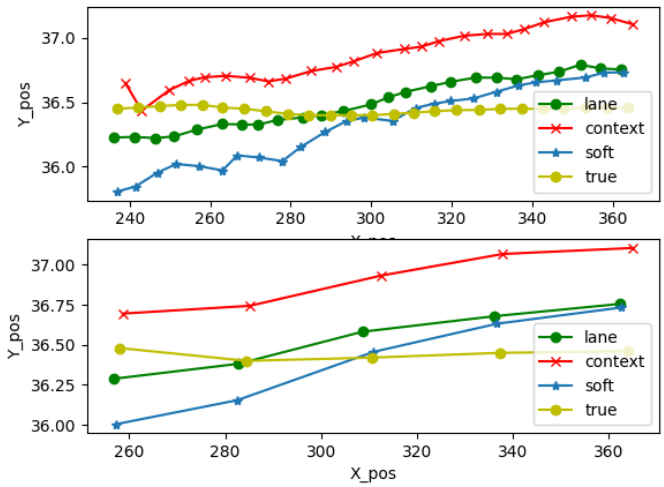


Figure 4.1: Comparison of predicted trajectory(lane keep)

In the lane change left trajectory, the models do not show accurate trajectories. The reason for this can be that the dataset is imbalanced. Lane keep scenarios are more compared to lane change scenarios. So, the quality of the predicted trajectory varies for each model. Also for both lane change left and lane change right, the accuracy of the context attention model is not accurate compared to the ground truth and also when comparing to lane attention model. Even though the context vector gives importance to the spatial information, the model finds it difficult to predict lane change scenarios.

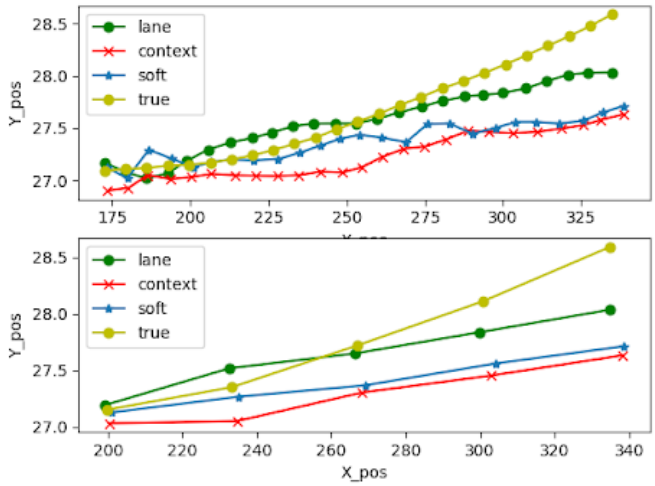


Figure 4.2: Comparison of predicted trajectory(lane change left)

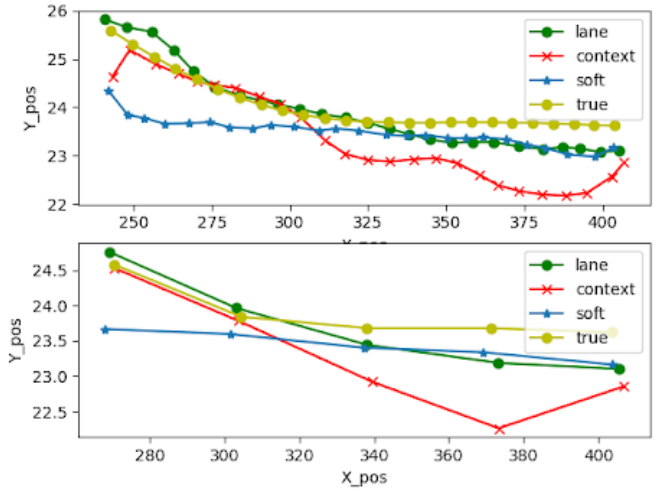


Figure 4.3: Comparison of predicted trajectory(lane change right)

4.2 Trajectory Prediction with Kalman Filter

As introduced in the previous chapter, Kalman filter is used to estimate the positions of the entire dataset before splitting the dataset into training, validation and testing sets. The ground truth is unfiltered data. The reason for choosing the ground truth as unfiltered is because this will give better predictions. Lane attention model seems to be the better model when compared with other models. It changes from 0.80 m in the 1st prediction horizon to 1.09 m in the 5th prediction horizon. This shows that it is better in long-term trajectory predictions. The soft attention model performance is not good even when compared with the baseline model in the 5th prediction horizon. This shows that the model is not good for long-term trajectory predictions. The main reason is that it does not give importance to the spatial information and only gives importance to the input and output sequences. Overall, the lane attention and context attention model perform better when the filter is used to estimate the positions.

Model	1s	2s	3s	4s	5s
Lane Attention	0.80	0.77	0.77	0.86	1.09
Context Attention	0.78	0.81	0.90	1.02	1.28
Soft Attention	1.29	1.29	1.38	1.51	1.89
Baseline	1.24	1.25	1.34	1.55	1.80

Table 4.4: Position RMSE(m)

In the longitudinal direction, lane attention performs better compared to the other models. It changes from 0.77 m to 1.03 m.

Model	1s	2s	3s	4s	5s
Lane Attention	0.77	0.74	0.74	0.81	1.02
Context Attention	0.75	0.78	0.86	0.97	1.32
Soft Attention	1.23	1.23	1.43	1.51	1.82
Baseline	1.18	1.20	1.29	1.49	1.77

Table 4.5: Longitudinal Position RMSE(m)

In the lateral direction, both the lane attention model and context attention model have equal results which is 0.35 m in the 5th prediction horizon. In this direction as well, the soft attention model performs lower than baseline model.

Model	1s	2s	3s	4s	5s
Lane Attention	0.21	0.22	0.22	0.28	0.35
Context Attention	0.21	0.21	0.23	0.28	0.35
Soft Attention	0.37	0.36	0.38	0.42	0.48
Baseline	0.40	0.38	0.37	0.41	0.49

Table 4.6: Lateral Position RMSE(m)

The predicted trajectories are plotted for the models against the ground truth for each instance of lane keep (fig 4.4), lane change left (fig 4.5) and lane change right (fig 4.6). In the lane keep trajectory, initially the context attention model is not close to the true trajectory. The trajectory of the models in lane keep scenario is accurate.

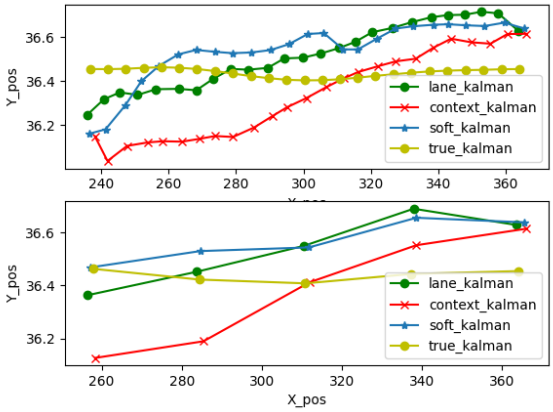


Figure 4.4: Comparison of predicted trajectory(lane keep)

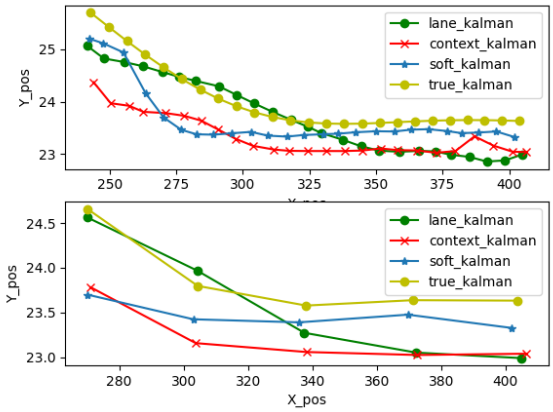


Figure 4.5: Comparison of predicted trajectory(lane change left)

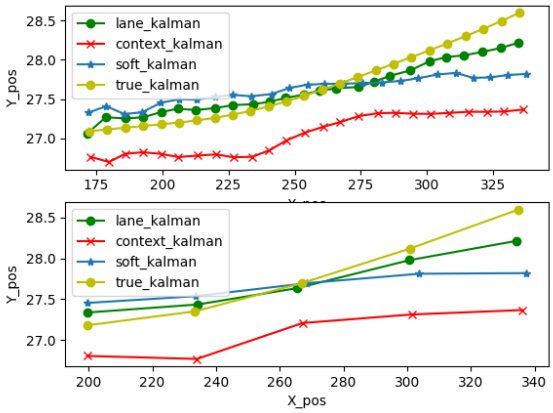


Figure 4.6: Comparison of predicted trajectory(lane change right)

4.3 Comparison between filtered and unfiltered models

The lane attention model and context attention model give better predictions when compared to the baseline model and soft attention model. The main reason is that the vehicles surrounding the target vehicle are given importance when the trajectory is predicted. When the filtered models and unfiltered models were compared, the lane attention and the context attention RMSE values looked better than the RMSE values from the unfiltered data. The RMSE value for the lane attention model in the 5th prediction horizon is 1.71 m with unfiltered data, while the RMSE value for the filtered data is 1.09 m. This shows that the more complicated models perform better when the filtered data is used. This also shows that it relies more on precise data to give accurate predictions. The RMSE values for the soft attention model and baseline model do not change significantly when filtered data is used. For both the unfiltered and filtered data, there is not a big change to the model when short-term prediction is done. Filtered data reduces the error when long-term prediction is done.

5
Conclusion and Future Work

5.1 Conclusion

In this thesis, an encoder-decoder LSTM framework has been used to predict the future of the target vehicle for a time horizon of 5 seconds. The network was trained and evaluated using the HighD dataset which is recorded during highway driving. Two approaches are proposed and compared using the same framework. The first approach is using a Kalman filter to the dataset to estimate the positions. The other approach is not to use a Kalman filter to the dataset. The approach using the Kalman filter shows better results compared to the unfiltered data when evaluating the models using RMSE values. The results are only better for the lane attention and context attention mechanism models. The RMSE values for the baseline and soft attention model do not show many changes when compared with unfiltered data. Lane attention model performs better compared to other models in cases where the target vehicle does a lane change.

Two attention mechanisms are implemented i.e. context attention and lane attention. The two attention mechanisms choose suitable context vectors for prediction. Generally, the two attention models i.e lane attention and context attention perform better when compared with the baseline model. The model with a lane attention mechanism performs better when compared with other models. This shows that lane attention is important in vehicle trajectory prediction. The soft attention mechanism model does not perform as well as the other two attention mechanism models.

Finally, the work in the thesis shows a potential method for making autonomous vehicles more aware of their surrounding vehicles and take decisions accordingly.

5.2 Future Work

The result of the thesis has only a particular driving scenario and virtual vehicles are also added to maintain the scenario. In the future, the methods could be replicated to different driving scenarios. More trajectory prediction data sets can also be implemented on the model to make the model more suitable to different traffic conditions. Also, by increasing the number of layers we can try changing the network structure and since, parameter tuning did not reach the best value, so this can be explored.

In this thesis, the positions are only predicted using the positions and velocity. A future work can be to add more features and try predicting the trajectory. The added features could be size of the vehicle, signal lights, etc. The vehicle trajectory dataset which is used is recorded using drones, so in future, data recorded from a single vehicle perspective can be used for trajectory prediction.

Bibliography

[1] Krajewski, Robert, et al. "The HighD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems." 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2118–25. IEEE Xplore, doi:10.1109/ITSC.2018.8569552.

[2] Ulbrich, Simon, and Markus Maurer. "Towards Tactical Lane Change Behavior Planning for Automated Vehicles." 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 989–95. IEEE Xplore, doi:10.1109/ITSC.2015.165.

[3] Skansi, Sandro. Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence. Springer, 2018.

[4] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long Short-Term Memory." Neural Computation, vol. 9, no. 8, Nov. 1997, pp. 1735–80. Silverchair, doi:10.1162/neco.1997.9.8.1735

[5] Georgiou, Harris, et al. "Moving Objects Analytics: Survey on Future Location Trajectory Prediction Methods." ArXiv:1807.04639 [Cs, Stat], July 2018. arXiv.org, http://arxiv.org/abs/1807.04639

[6] Lefèvre, Stéphanie, et al. "A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles." ROBOMECH Journal, vol. 1, no. 1, July 2014, p. 1. BioMed Central, doi:10.1186/s40648-014-0001-z

[7] Schubert, Robin, et al. "Comparison and Evaluation of Advanced Motion Models for Vehicle Tracking." 2008 11th International Conference on Information Fusion, 2008, pp. 1–6

[8] Schlechtriemen, Julian, et al. "When Will It Change the Lane? A Probabilistic Regression Approach for Rarely Occurring Events." 2015 IEEE Intelligent Vehicles Symposium (IV), 2015, pp. 1373–79. IEEE Xplore, doi:10.1109/IVS.2015.7225907

[9] Ammoun, Samer, and Fawzi Nashashibi. "Real Time Trajectory Prediction for Collision Risk Estimation between Vehicles." 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing, 2009, pp. 417–22. IEEE Xplore, doi:10.1109/ICCP.2009.5284727.

[10] Hadfield-Menell, Dylan, et al. "Cooperative Inverse Reinforcement Learning." ArXiv:1606.03137 [Cs], Nov. 2016. arXiv.org, http://arxiv.org/abs/1606.03137

[11] S. Yoon and D. Kum, "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles," in 2016 IEEE Intelligent Vehicles Symposium (IV), June 2016, pp. 1307–1312

[12] P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in 2013 IEEE Intelligent Vehicles Symposium (IV), June 2013, pp. 797–802

[13] Meyer-Delius, Daniel, et al. "Probabilistic Situation Recognition for Vehicular Traffic Scenarios." 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 459–64. IEEE Xplore, doi:10.1109/ROBOT.2009.5152838

[14] J. Wiest, M. Höffken, U. Kressel, and K. Dietmayer, "Probabilistic trajectory prediction with gaussian mixture models," in 2012 IEEE Intelligent Vehicles Symposium (IV), June 2012, pp. 141–146

[15] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. "A survey on motion prediction and risk assessment for intelligent vehicles". In: ROBOMECH journal 1.1 (2014), p. 1

[16] ByeoungDo Kim et al. "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network". In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE. 2017, pp. 399–404

[17] Nal Kalchbrenner and Phil Blunsom. "Recurrent continuous translation models". In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013, pp. 1700–1709

[18] Deo, N., Rangesh, A., Trivedi, M.M., et al.: 'How would surround vehicles move? a unified framework for maneuver classification and motion prediction', IEEE Trans. Intell. Veh., 2018, 3, (2), pp. 129–140

[19] Lawitzky, A., Althoff, D., Passenberg, C.F., et al.: 'Interactive scene prediction for automotive applications'. IEEE Int. Conf. Intelligent Vehicles Symp., Gold Coast, Australia, June 2013, pp. 1028–1033

[20] Deo, N., Trivedi, M.M.: 'Convolutional social pooling for vehicle trajectory prediction'. IEEE Int. Conf. Computer Vision and Pattern Recognition Workshops, Salt Lake City, USA, June 2018, pp. 1468–1476

[21] Massaoud, K., Yahiaoui, I., Verroust-blondet, A., et al.: 'Non-local social pooling for vehicle trajectory prediction'. Int. Conf. Intelligent Vehicles Symp., Paris, France, June 2019, pp. 975–980

[22] Bahdanau, Dzmitry, et al. "Neural Machine Translation by Jointly Learning to Align and Translate." ArXiv:1409.0473 [Cs, Stat], May 2016. arXiv.org, http://arxiv.org/abs/1409.0473.

[23] "Soft + Hardwired Attention: An LSTM Framework for Human Trajectory Prediction and Abnormal Event Detection." Neural Networks, vol. 108, Dec. 2018, pp. 466–78. www.sciencedirect.com, doi:10.1016/j.neunet.2018.09.002

[24] van Gerven, Marcel, and Sander Bohte. "Editorial: Artificial Neural Networks as Models of Neural Information Processing." Frontiers in Computational Neuroscience, vol. 11, 2017. Frontiers, doi:10.3389/fncom.2017.00114.

[25] Bilal, Afaan. "Artificial Neural Networks and Deep Learning." Medium, 8 Feb. 2018, https://becominghuman.ai/artificial-neural-networks-and-deep-learning-a3c9136f2137

[26] "The Artificial Neural Networks Handbook: Part 4 - DZone AI." Dzone.Com, https://dzone.com/articles/the-artificial-neural-networks-handbook-part-4. Accessed 13 May 2021

[27] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org

[28] SHARMA, SAGAR. "Activation Functions in Neural Networks." Medium, 14 Feb. 2019, https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[29] Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." ArXiv:1412.6980 [Cs], Jan. 2017. arXiv.org, http://arxiv.org/abs/1412.6980

[30] Skansi, Sandro. Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence. Springer, 2018.

[31] Bahdanau, Dzmitry, et al. "Neural Machine Translation by Jointly Learning to Align and Translate." ArXiv:1409.0473 [Cs, Stat], May 2016. arXiv.org, http://arxiv.org/abs/1409.0473.

[32] "Attention Mechanism In Deep Learning | Attention Model Keras." Analytics Vidhya, 20 Nov. 2019, https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/.

[33] HighD Dataset Format. https://www.ika.rwth-aachen.de/en/723-drone-datasets/3316-highd-format.html. Accessed 28 May 2021.

[34] Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data - CKAN. https://catalog.data.gov/dataset/next-generation-simulation-ngsim-vehicle-trajectories-and-supporting-data. Accessed 3 June 2021.

[35] Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." ArXiv:1412.6980 [Cs], Jan. 2017. arXiv.org, http://arxiv.org/abs/1412.6980.

[36] Yan, Jun, et al. "Trajectory Prediction for Intelligent Vehicles Using Spatial-Attention Mechanism." IET Intelligent Transport Systems, vol. 14, no. 13, 2020, pp. 1855–63. Wiley Online Library, doi:https://doi.org/10.1049/iet-its.2020.0274.

[37] Motion Prediction of Surrounding Vehicles in Highway Scenarios with Deep Learning, Samir Khays, Master Thesis Report

[38] Yiru Lyu,Behavior Prediction of Surrounding Vehicles in a Road Network for Autonomous Driving, Master Thesis Report

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY